

User Interface Design

(source: Rajib Mall)

Anil Kumar Dudyala

Asst.Prof, Dept of CSE
NIT Patna



Introduction

- In any software product,
 - user interface portion is responsible for all interactions with the user.
- Almost every software product has a user interface:
 - can you think of any software product that does not have any user interface?



Introduction

- In the early days of computer, no software product had any user interface:
 - all computers were batch systems
 - no interactions with users were supported.
- We know that things are very different now:
 - almost every software product is highly interactive.



Introduction

- Users interact with a software product through its user interface:
 - user-interface portion of any software is directly relevant to the users.
 - many users judge a software from its user interface.
- User interface design:
 - a practical and important problem.



Introduction

- Aesthetics apart, a difficult to use interface:
 - leads to higher levels of user errors
 - leads to user dissatisfaction.
- Users become particularly irritated when a system behaves in unexpected ways,
 - issued commands do not carry out actions according to intuitive expectations of users.



Introduction

- A significant portion of the total development effort:
 - spent in developing the user interface.
- For many interactive applications:
 - as much as 50% of the total development effort is spent on the user interface part.



Introduction

- If the user interface is not developed systematically:
 - effort required to develop the interface would increase tremendously :
 - leading to customer dissatisfaction.



Introduction

- It is important to carefully study
 - user interface design concepts.
- Let us examine what a good interface is:
 - Unless we know what a good user interface really is,
 - we cannot possibly design one.



Characteristics of Good User Interfaces

- Speed of learning:
 - A good user interface should be simple to learn.
 - Most users are put off by complex syntax and semantics of the command issue procedures.



Speed of learning

- A good user interface should not require
 - users to memorize commands.
- An important factor affecting the speed of learning is consistency:
 - Once, a user learns about a command,
 - should be able to use similar commands
 - even in different circumstances for carrying out similar actions.



Speed of learning

- Users can learn about an interface faster, if it is based on:
 - day-to-day real life examples (aka metaphors)
 - use concepts with which users are already familiar.



Speed of learning

- For example, interface of a text editor:
 - can use concepts similar to writing on paper:
 - such as cutting lines and paragraphs and pasting it at other places,
 - users can immediately relate to it.
- Also, learning is facilitated by:
 - intuitive command names
 - symbolic command issue procedures.



Speed of use

- Speed of use is determined by:
 - the time and effort necessary to initiate and execute different commands.
- The time and user effort necessary to execute different commands should be minimal.



Speed of use

- Examples of bad interfaces:
 - users required to type in lengthy commands
 - command issue involves moving the mouse to widely different areas of the screen
 - can slow down the operating speed of users.



Speed of recall

- Once users learn how to use an interface:
 - their speed of recall about using the interface should be maximized.
- The speed of recall is improved if the interface is based on:
 - metaphors
 - symbolic command issue procedures
 - intuitive command names.



Error rate

- A good user interface should
 - minimize the scope of committing errors.
- Error rate can be easily measured:
 - count the errors committed by different users.



Error rate

- Error monitoring can be automated:
 - instrument user interface with monitoring code
 - record the frequency and types of errors committed by different users
 - later display statistics of various kinds of user errors.



How to reduce error possibilities?

- Consistency of command names,
- Consistency of command issue procedures,
- Consistency in behavior of similar commands
- Simplicity of command issue procedure, etc.



Attractiveness

- A good user interface should be attractive:
 - An attractive user interface catches user attention and fancy.
- In this respect,
 - graphics-based user interfaces have a definite advantage over text-based interfaces.



Consistency

- Consistency of commands is very desirable.
 - allow users to generalize the knowledge about one aspect of the interface to another.
- Consistency helps in:
 - speed of learning,
 - speed of recall,
 - also helps in reduction of error rate.



Feedback

- A good user interface must provide constant feedback to user actions:
 - For example, if any user request takes more than few seconds to process,
 - the user must be informed that his/her request is still being processed.
- If possible, the user should be periodically informed about the progress made in processing the command.



Feedback

- In the absence of any response from the computer for a long time:
 - a novice user might even start recovery/shutdown procedures in panic.



Support for multiple skill levels

- A good user interface:
 - should support different levels of sophistication in command issue procedures:
 - users with different experience levels prefer different types of interfaces.
- Experienced users are more concerned about speed of command issue:
 - whereas novice users pay prime importance to usability aspects.



Support for multiple skill levels

- Novice users discouraged by:
 - cryptic command names
 - complex command issue procedures.
- Elaborate commands:
 - slow down command issue procedure
 - put off experienced users.



Support for multiple skill levels

- As users become familiar with an interface:
 - look for faster command issue procedures such as "hot-keys", "macros", etc.



Error Recovery (Undo facility)

- All categories of users commit errors.
 - A good interface should allow users to undo mistakes..
- Users are inconvenienced:
 - if they can not recover from even simple errors.



User Guidance and On-line Help

- Users might need guidance
 - or seek help from the system.
- User Guidance is provided through two broad category of methods:
 - On-line help system
 - Guidance and error messages produced
 - in response to user actions.



On-line Help System

- Generic help messages are not very useful:
 - on-line help messages should be tailored to the context in which help is invoked.



On-line Help

- A good on-line help should:
 - provide messages in context-dependent way.
 - keep track of what a user is doing
 - help messages should be tailored to user's experience level.
- should take advantage of graphics capabilities of the screen
 - not just be a copy of the user manual.



Guidance Messages

- The guidance messages should be carefully designed:
 - prompt the user:
 - next actions he/she might take,
 - current status of the system,
 - progress made so far in processing the command



Guidance Messages

- A good guidance system should have:
 - different levels of sophistication for different categories of users.
 - users should have an option to turn off detailed messages.



Error Messages

- Error messages should be polite.
- Error messages should not be associated with noise:
 - might embarrass the user.
- The messages should suggest how a given error can be rectified.



Error Messages

- If appropriate,
 - the user should be given the option of invoking on-line help
- to find out more about the error situation.



Improper Error Message



File not found!



Mode-Based versus Modeless Interface

- A mode is a state or collection of states:
 - in each state (or mode) different types of commands are available.
- Modeless interface:
 - same set of commands are available at all times.
- Mode-based interface:
 - different sets of commands are available depending on the mode in which the system is, i.e.
 - based on the past sequence of the user commands.



Mode-Based versus Modeless Interface

- A mode-based interface can be represented using a state transition diagram:
 - each node (state) of the state transition diagram represents a mode.
- Every state (node) of the diagram
 - annotated with command names meaningful in that state.



Graphical User Interface (GUI) versus Text-Based User Interface

- In a GUI:
 - several windows with different information can be simultaneously displayed on user's screen.
 - This is perhaps the biggest advantage of GUI
 - user can simultaneously interact with several related items at any time
 - can even run many unrelated applications



Graphical User Interface (GUI) versus Text-Based User Interface

- Iconic information representation and symbolic information manipulation is possible in a GUI.
- Symbolic information manipulation:
 - such as pulling an icon representing a file into a trash can for deleting
 - intuitively very appealing
 - user can instantly remember it.



Graphical User Interface (GUI) versus Text-Based User Interface

- A GUI can support command selection:
 - using an attractive and user-friendly menu selection system.
- In a GUI, a pointing device can be used:
 - a mouse or a light pen to issue commands.
 - The use of a pointing device
 - makes command issue procedure simpler.



Graphical User Interface (GUI) versus Text-Based User Interface

- On the flip side, a GUI requires:
 - special terminals with graphics capabilities
 - requires special input devices such a mouse, touch screens, etc.
- In contrast, a text-based interface:
 - can run even on cheap alphanumeric terminals:
 - Graphics terminals are usually much more expensive than alphanumeric terminals.



Graphical User Interface (GUI) versus Text-Based User Interface

- Increasing availability of:
 - terminals with graphics capability
 - bit-mapped high-resolution displays
 - significant amount of local processing power.
- We will concentrate our attention to GUIs



Types of User Interfaces

- User interfaces can be classified into three categories:
 - Command language-based interface
 - Menu-based interface
 - Direct manipulation interface



Types of User Interfaces

- Each category of interface has its advantages and disadvantages:
 - Modern applications sport a combination of all the three types of interfaces.



Choice of Interface

- Which parts of the interface should be implemented using what type of interface?
 - No simple guidelines available
- to a large extent depends on the experience and discretion of the designer.
- a study of characteristics of the different interfaces would give us some idea.



Command Language-Based Interface

- As the name itself suggests:
 - incorporates some language to form commands.
- Users frame the required commands in the language:
 - type them in whenever required.



Design of command language interface

- Simple command language interface:
 - determine all the commands needed to be supported
 - assign unique names to the different commands.



Design of command language interface

- A more sophisticated command language interface:
 - allow users to compose primitive commands to form more complex commands.
 - Consider `cat x.dat|grep 123`
- Like a programming language.



Command Language-Based Interface

- The facility to compose commands:
 - dramatically reduces the number of command names users would have to remember.
- Commands can be made concise
 - requiring minimal typing by the user.
- allow faster interaction with the computer
- simplify input of complex commands.



Advantages of Command Language Interfaces

- Easy to develop:
 - compiler writing techniques are well developed.
- Can be implemented even on cheap alphanumeric terminals.
- Much more efficient:
 - compared to other types of interfaces.



Disadvantages of Command Language Interfaces

- Difficult to learn:
 - Require the user to memorize primitive commands.
- Require the user to type in commands.
- Users make errors while:
 - formulating commands in the command language
 - typing them in.



Disadvantages of Command Language Interfaces

- All interactions are through key-board:
 - cannot take advantage of effective interaction devices such as a mouse.
- For casual and inexperienced users,
 - command language interfaces are not suitable.



Issues in Designing a Command Language Interface

- Design of a command language interface:
 - involves several issues.
- The designer has to decide
 - what **mnemonics** are to be used for the commands.
 - mnemonics should be meaningful
 - yet be concise to minimize the amount of typing required.



Issues in Designing a Command Language Interface

- The designer has to decide:
 - whether users will be allowed to redefine command names to suit their own preferences.
 - Letting a user define his own mnemonics for various commands is a useful feature,
 - but increases complexity of user interface development.



Issues in Designing a Command Language Interface

- Designer has to decide:
 - whether it should be possible to compose primitive commands to create more complex commands.
 - syntax and semantics of command composition options has to be clearly and unambiguously decided.



Issues in Designing a Command Language Interface

- The ability to combine commands is a powerful facility for experienced users:
 - but quite unnecessary for inexperienced users.



Menu-Based Interface

- Advantages of a menu-based interface over a command language interface:
 - users are not required to remember exact command names.
- typing effort is minimal:
 - menu selections using a pointing device.
 - This factor becomes very important for the occasional users who can not type fast.



Menu-Based Interface

- For experienced users:
 - menu-based interfaces is slower than command language interfaces
 - experienced users can type fast
 - also get speed advantage by composing simple commands into complex commands.



Menu-Based Interface

- Composition of commands in a menu-based interface is not possible.
 - actions involving logical connectives (and, or, all, etc.)
 - awkward to specify in a menu-based system.



Menu-Based Interface

- If the number of choices is large,
 - it is difficult to design a menu-based interface.
 - Even moderate sized software needs hundreds or thousands of menu choices.
- A major problem with the menu-based interface:
 - structuring large number of menu choices into manageable forms.



Structuring Menu Interface

- Any one of the following options is adopted to structure menu items.
 - Walking menu
 - Scrolling menu
 - Hierarchical menu



Scrolling Menu

- Used when the menu options are highly related.
 - For example text height selection in a word processing software.
- Scrolling of menu items
 - lets the user to view and select the menu items that can not be accommodated on one screen.



Walking Menu

- Walking menu is commonly used to structure large menu lists:
- when a menu item is selected, it causes further menu items to be displayed adjacent to it in a submenu.



Walking Menu

- A walking menu can successfully structure commands only if:
 - there are tens rather than hundreds of choices
- each adjacently displayed menu does take up some screen space
 - the total screen area is after all limited.



Hierarchical menu

- Menu items are organized in a hierarchy or tree structure.
 - Selecting a menu item causes the current menu display to be replaced by an appropriate submenu.
- One can consider the menu and its various submenu to form a hierarchical tree-like structure.



Hierarchical menu

- Walking menu are a form of hierarchical menu:
 - practicable when the tree is shallow.
- Hierarchical menu can be used to manage large number of choices,
 - but, users face navigational problems
 - lose track of where they are in the menu tree.



Direct Manipulation Interface

- Present information to the user
 - as **visual models or objects**.
- Actions are performed on the visual representations of the objects, e.g.
 - pull an icon representing a file into an icon representing a trash box, for deleting the file.
- Direct manipulation interfaces are sometimes called as **iconic interfaces**.



Direct Manipulation (Iconic) Interface

- Important advantages of iconic interfaces:
 - icons can be recognized by users very easily,
 - icons are language-independent.
- However, experienced users consider direct manipulation interfaces too slow.



Direct Manipulation (Iconic) Interface

- It is difficult to form complex commands using a direct manipulation interface.
- For example, if one has to drag a file icon into a trash box icon for deleting a file:
 - to delete all files in a directory one has to perform this operation again and again
 - very easily done in a command language-interface by issuing a command `delete *.*`



Windowing Systems

- Most modern GUIs are developed using some [windowing system](#).
- A windowing system can generate displays through a set of windows.
- Since a window is a basic entity in such a graphical user interface:
 - we need to first discuss what exactly a window is.



Window

- A window is a rectangular area on the screen.
- A window is a virtual screen:
 - it provides an interface to the user for carrying out independent activities,
 - one window can be used for editing a program and another for drawing pictures, etc.
- A window can be divided into two parts:
 - client part,
 - non-client part.



Window

- The client area makes up the whole of the window,
 - except for the borders and scroll bars.
- The client area is available to the programmer.
- Non-client area:
 - under the control of window manager.



Window management system (WMS)

- A graphical interface might consist of a large number of windows,

necessary to have some systematic way to manage the windows.

Window Management System (WMS)



Window management system (WMS)

- GUIs are developed using a window management system (WMS):
 - A window management system is primarily a resource manager.
- keeps track of screen area resource
- allocates it to the different windows which are using the screen.



Window management system (WMS)

- From a broader perspective, a WMS can be considered as a **user interface management system (UIMS)** ---
 - not only does resource management,
 - also provides the basic behavior to windows
 - provides several utility routines to the application programmer for user interface development.



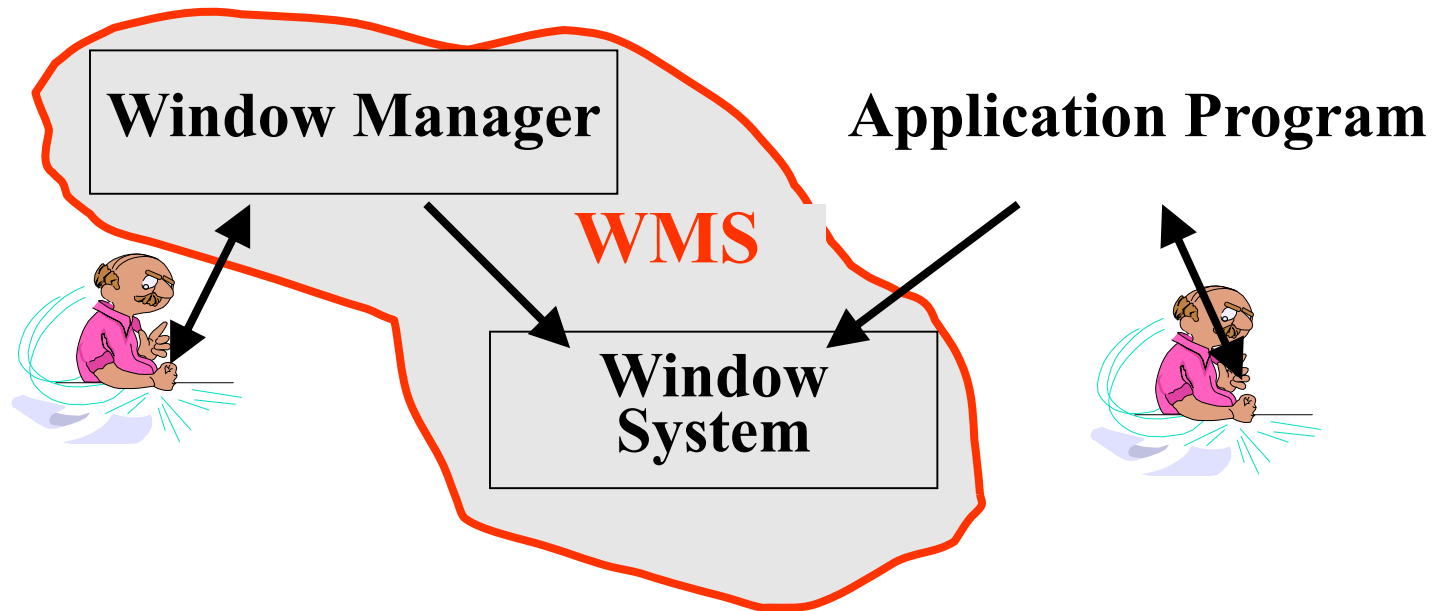
Window management system (WMS)

- A WMS simplifies the task of a GUI designer to a great extent:
 - provides the basic behavior to the various windows such as move, resize, iconify, etc.
- provides routines to manipulate windows such as:
 - creating, destroying, changing attributes of the windows, and drawing text, lines, etc.



Window management system (WMS)

- A WMS consists of two part:
 - a window manager
 - a window system.



Window Manager and Window System

- User interacts with window manager to do various window-related operations such as:
 - window repositioning,
 - window resizing,
 - iconification, etc.
- Window manager also controls the screen's real estate policy.



Window manager

- The window manager is built on the top of the window system:
 - makes use of the basic services provided by the window system.
- The window manager determines how the windows look and behave.
 - several kinds of window managers can be based on the same window system.



Window manager

- Window manager can be considered as a special program:
 - makes use of the services (function calls) of the window system.
- Application programs
 - invoke the window system for user interface-related functions.



Window System

- Provides a large number of routines for the programmer
- It is very cumbersome to use these large set of routines:
 - most WMS provide a higher-level abstraction called [widgets](#).



Window Management System

- A widget is the short form for a [window object](#).
- Widgets are the building blocks in interface design.
- We know that an object is essentially a collection of:
 - related data with several operations defined on these data.



Widgets

- The data of an window object are:
 - the geometric attributes (such as size, location etc.)
 - other attributes such as its background and foreground color, etc.
- The operations defined on these data include, resize, move, draw, etc.



Advantages of Widgets

- One of the most important reasons to use widgets as building blocks:
 - provide consistency.
- Consistent user interfaces
 - improve the user's productivity and
 - lead to higher performance with fewer errors.



Advantages of Widgets

- Widgets make users familiar with standard ways of using an interface ---
 - users can easily extend their knowledge of interface of one application to another
- the learning time for users is reduced to a great extent.



Summary

- User interface is an important component of any software product.
- We first discussed some important concepts associated with user interface design.
- We discussed some desirable properties of a good user interface.



Summary

- There are 3 main types of interfaces:
 - command language interface
 - menu-based interface
 - iconic interface
- Each type of interface has its own advantages and disadvantages:
 - most modern interfaces use a combination of all the three types.



Summary

- We finally discussed concepts associated with a window management system (WMS):
 - consists of a window manager and a window system.
 - we discussed the use of widgets as the basic building blocks in GUI design.

